Multicollinearity

March 12, 2024

1 Multicollinearity

Multicollinearity is a situation in regression analysis where independent variables are highly correlated. This can cause problems in estimating the coefficients of the regression model, as it becomes difficult to isolate the individual effect of each predictor. The high correlation inflates the standard errors of the coefficients, making it challenging to determine the significance of the predictors.

The Variance Inflation Factor (VIF) is a common measure used to detect multicollinearity. The VIF for a predictor variable is calculated as:

$$VIF = \frac{1}{1 - R^2}$$

where R^2 is the coefficient of determination obtained by regressing the predictor variable against all other predictor variables. A VIF value greater than 5 or 10 indicates a problematic level of multicollinearity.

This equation, known as auxiliary regression is of the form:

$$X_1 = \beta_0 + \beta_2 X_2 + \beta_3 X_3 + \epsilon$$

In this auxiliary regression, X_1 is treated as the dependent variable, and the other independent variables X_2 and X_3 are treated as the independent variables. The coefficients β_2 and β_3 represent the impact of X_2 and X_3 on X_1, respectively, and *epsilon* is the error term.

The VIF for X_1 is then calculated as:

$$VIF(X_1) = \frac{1}{1 - R^2_{X_1}}$$

where $R^2_{X_1}$ is the coefficient of determination (R-squared) from the auxiliary regression of X_1 on X_2 and X_3. This process is repeated for each independent variable in the model to calculate their respective VIFs.

To solve multicollinearity, you can consider the following approaches:
1. Remove highly correlated independent variables.
2. Combine linearly related variables.
3. Use partial least squares regression or principal component analysis to create uncorrelated components.

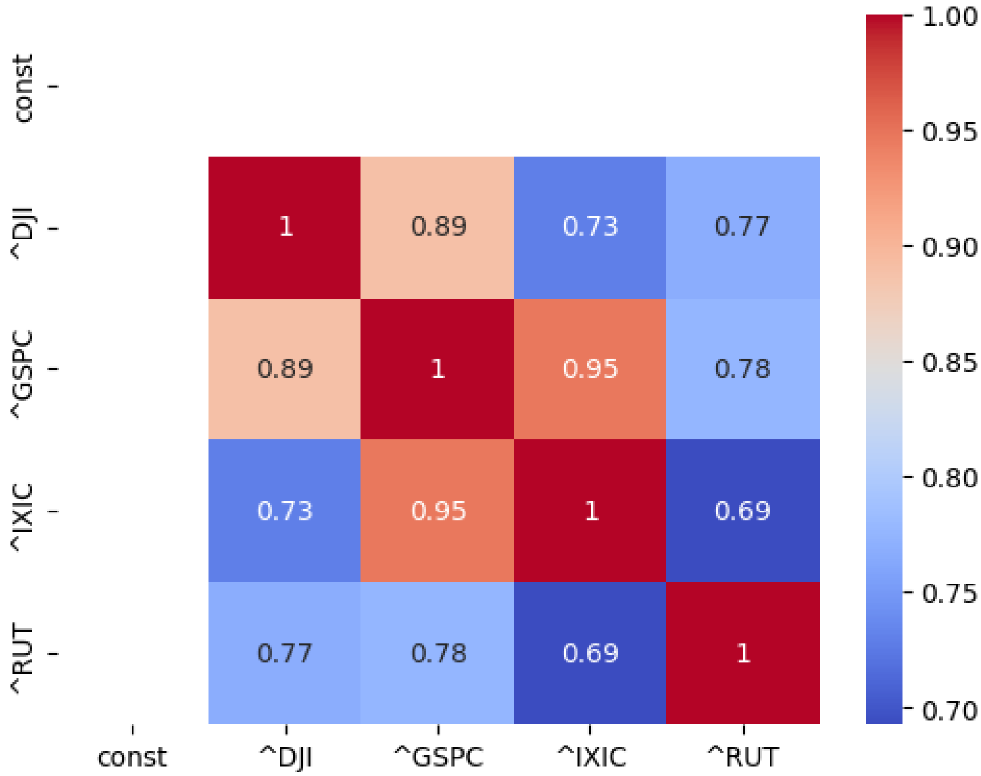
4. Employ regularization techniques like LASSO or Ridge regression, which can handle multicollinearity by penalizing large coefficients.

2 Exercise to test and solve (if needed) Multicollinearity

[3]:
```
import pandas as pd
import numpy as np
import yfinance as yf
import statsmodels.api as sm
import seaborn as sns
import matplotlib.pyplot as plt
from statsmodels.stats.outliers_influence import variance_inflation_factor

# Download data from Yahoo Finance
start_date = '2023-01-01'
end_date = pd.to_datetime('today').strftime('%Y-%m-%d')
tickers = ['NVDA', '^GSPC', '^RUT', '^IXIC', '^DJI']
data = yf.download(tickers, start=start_date, end=end_date)['Adj Close']

# Calculate daily returns
returns = data.pct_change().dropna()

# Define dependent and independent variables
y = returns['NVDA']
X = returns.drop(columns=['NVDA'])
X = sm.add_constant(X)  # Add a constant term to the model

# Fit the CAPM model
model = sm.OLS(y, X)
results = model.fit()
print(results.summary())

# Plot correlation matrix
sns.heatmap(X.corr(), annot=True, cmap='coolwarm')
plt.show()

# Calculate VIF for each independent variable
vif_data = pd.DataFrame()
vif_data['variable'] = X.columns
vif_data['VIF'] = [variance_inflation_factor(X.values, i) for i in range(X.
 ↪shape[1])]
print(vif_data)

# Check for high VIF and correct if necessary
high_vif = vif_data[vif_data['VIF'] > 5]
if not high_vif.empty:
```

```python
    print("High VIF detected. Consider removing or transforming variables.")
    # Example correction: Remove the variable with the highest VIF
    # X = X.drop(columns=[high_vif['variable'].iloc[0]])
    # Re-fit the model and re-calculate VIF if necessary
else:
    print("No high VIF detected.")
```

```
[*********************100%%**********************]  5 of 5 completed
                            OLS Regression Results
==============================================================================
Dep. Variable:                   NVDA   R-squared:                       0.534
Model:                            OLS   Adj. R-squared:                  0.528
Method:                 Least Squares   F-statistic:                     83.62
Date:                Tue, 12 Mar 2024   Prob (F-statistic):           3.14e-47
Time:                        00:37:25   Log-Likelihood:                 723.64
No. Observations:                 297   AIC:                            -1437.
Df Residuals:                     292   BIC:                            -1419.
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0031      0.001      2.465      0.014       0.001       0.006
^DJI          -2.4559      0.602     -4.083      0.000      -3.640      -1.272
^GSPC          2.6400      1.073      2.461      0.014       0.529       4.751
^IXIC          1.5932      0.532      2.995      0.003       0.546       2.640
^RUT          -0.5064      0.160     -3.162      0.002      -0.822      -0.191
==============================================================================
Omnibus:                      238.025   Durbin-Watson:                   2.145
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             6626.979
Skew:                           2.958   Prob(JB):                         0.00
Kurtosis:                      25.372   Cond. No.                     1.05e+03
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The condition number is large, 1.05e+03. This might indicate that there are
strong multicollinearity or other numerical problems.
```



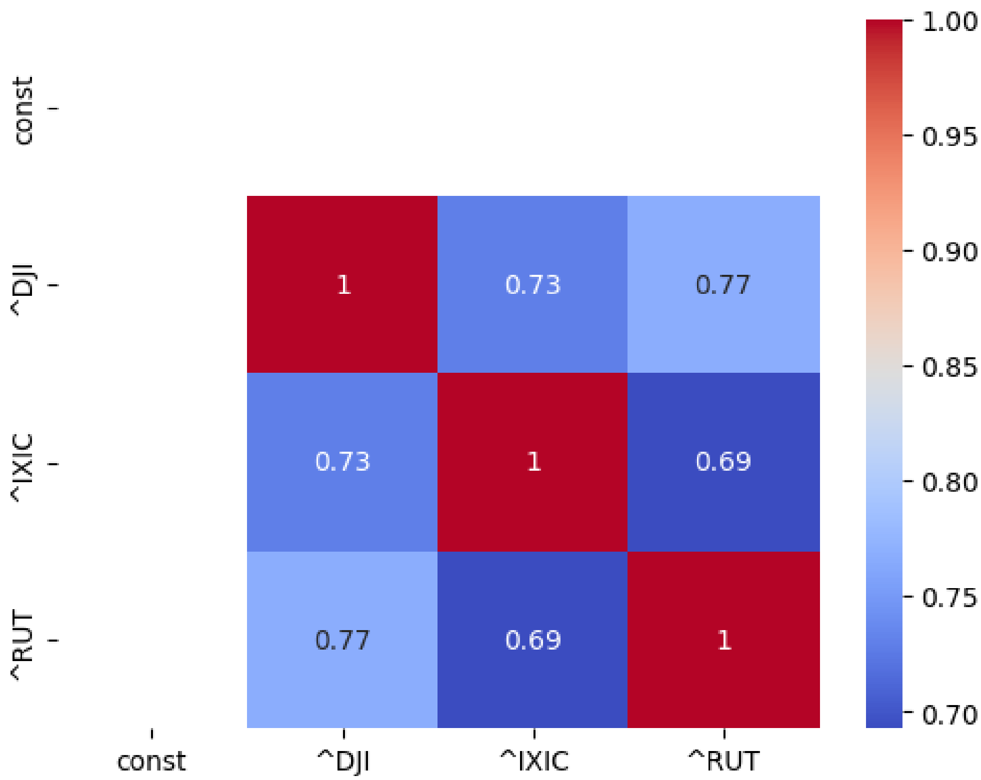

```
  variable        VIF
0    const   1.024237
1     ^DJI  11.227818
2    ^GSPC  49.002078
3    ^IXIC  21.825885
4     ^RUT   2.710733
High VIF detected. Consider removing or transforming variables.
```

Removing S&P500

[4]:
```
# Define dependent and independent variables
y = returns['NVDA']
X = returns.drop(columns=['NVDA', '^GSPC'])  # Remove S&P 500
X = sm.add_constant(X)  # Add a constant term to the model

# Fit the CAPM model
model = sm.OLS(y, X)
results = model.fit()
print(results.summary())

# Plot correlation matrix
sns.heatmap(X.corr(), annot=True, cmap='coolwarm')
plt.show()
```

```python
# Calculate VIF for each independent variable
vif_data = pd.DataFrame()
vif_data['variable'] = X.columns
vif_data['VIF'] = [variance_inflation_factor(X.values, i) for i in range(X.
 ↪shape[1])]
print(vif_data)
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                   NVDA   R-squared:                       0.524
Model:                            OLS   Adj. R-squared:                  0.519
Method:                 Least Squares   F-statistic:                     107.6
Date:                Tue, 12 Mar 2024   Prob (F-statistic):           5.46e-47
Time:                        00:37:26   Log-Likelihood:                 720.59
No. Observations:                 297   AIC:                            -1433.
Df Residuals:                     293   BIC:                            -1418.
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const          0.0032      0.001      2.503      0.013       0.001       0.006
^DJI          -1.1874      0.313     -3.797      0.000      -1.803      -0.572
^IXIC          2.8296      0.176     16.072      0.000       2.483       3.176
^RUT          -0.4661      0.161     -2.900      0.004      -0.782      -0.150
==============================================================================
Omnibus:                      242.904   Durbin-Watson:                   2.145
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             7081.731
Skew:                           3.034   Prob(JB):                         0.00
Kurtosis:                      26.139   Cond. No.                         267.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
```




```
  variable       VIF
0    const  1.023661
1     ^DJI  2.983107
2    ^IXIC  2.350743
3     ^RUT  2.682420
```

All VIFs are lower than 5. -> The multicollinearity issue has been solved.

[6]:
```
!sudo apt-get install texlive-xetex texlive-fonts-recommended
 ↪texlive-plain-generic > /dev/null 2>&1
!jupyter nbconvert --to pdf /content/drive/MyDrive/Econ_Models/
 ↪Multicollinearity.ipynb
```

```
[NbConvertApp] Converting notebook
/content/drive/MyDrive/Econ_Models/Multicollinearity.ipynb to pdf
[NbConvertApp] Support files will be in Multicollinearity_files/
[NbConvertApp] Making directory ./Multicollinearity_files
[NbConvertApp] Making directory ./Multicollinearity_files
[NbConvertApp] Writing 43711 bytes to notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: ['xelatex', 'notebook.tex', '-quiet']
[NbConvertApp] Running bibtex 1 time: ['bibtex', 'notebook']
[NbConvertApp] WARNING | bibtex had problems, most likely because there were no
```

```
citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 100345 bytes to
/content/drive/MyDrive/Econ_Models/Multicollinearity.pdf
```